

1/23

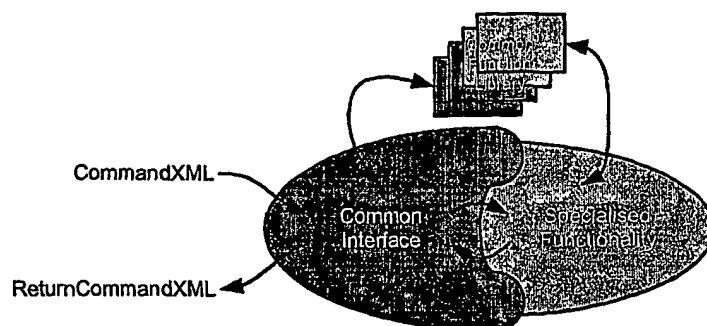


Figure 1 – Data Collector System Object

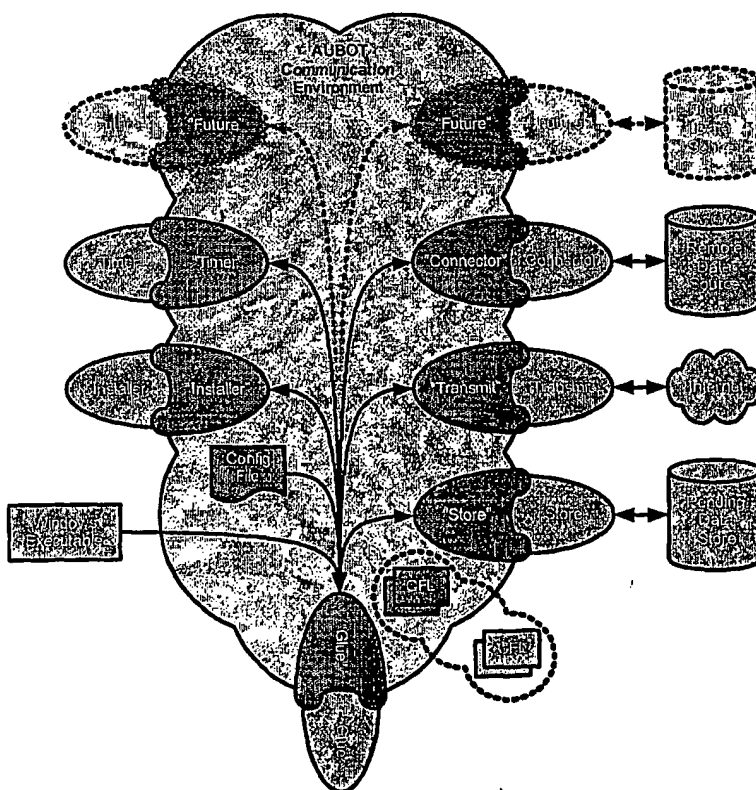


Figure 2- Data Collector System Communication Environment

2/23

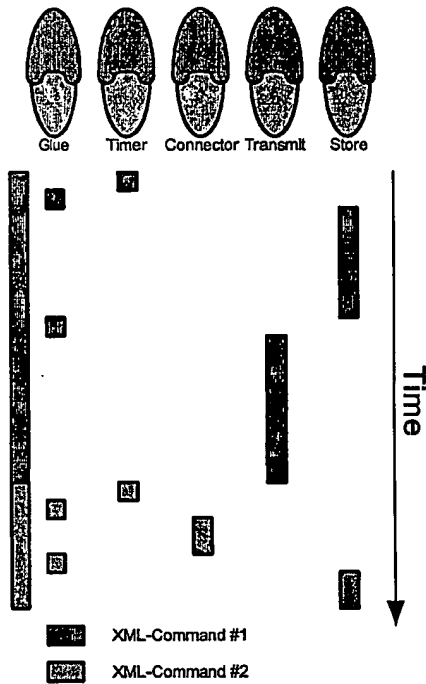


Figure 3 - Synchronous

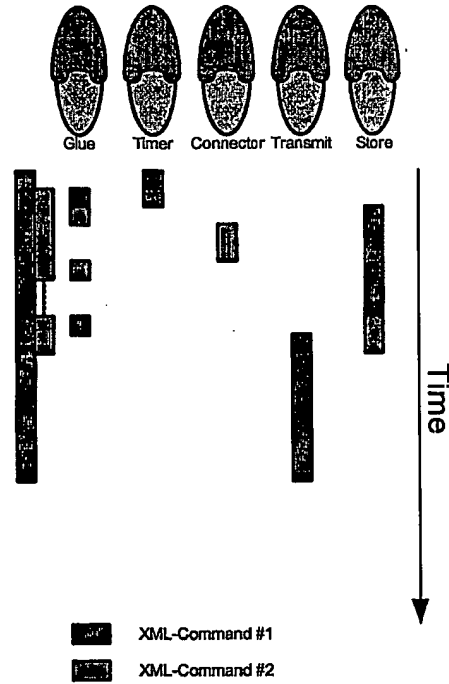


Figure 4 - Asynchronous

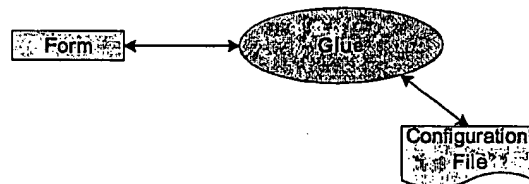


Figure 5 Core system

3/23

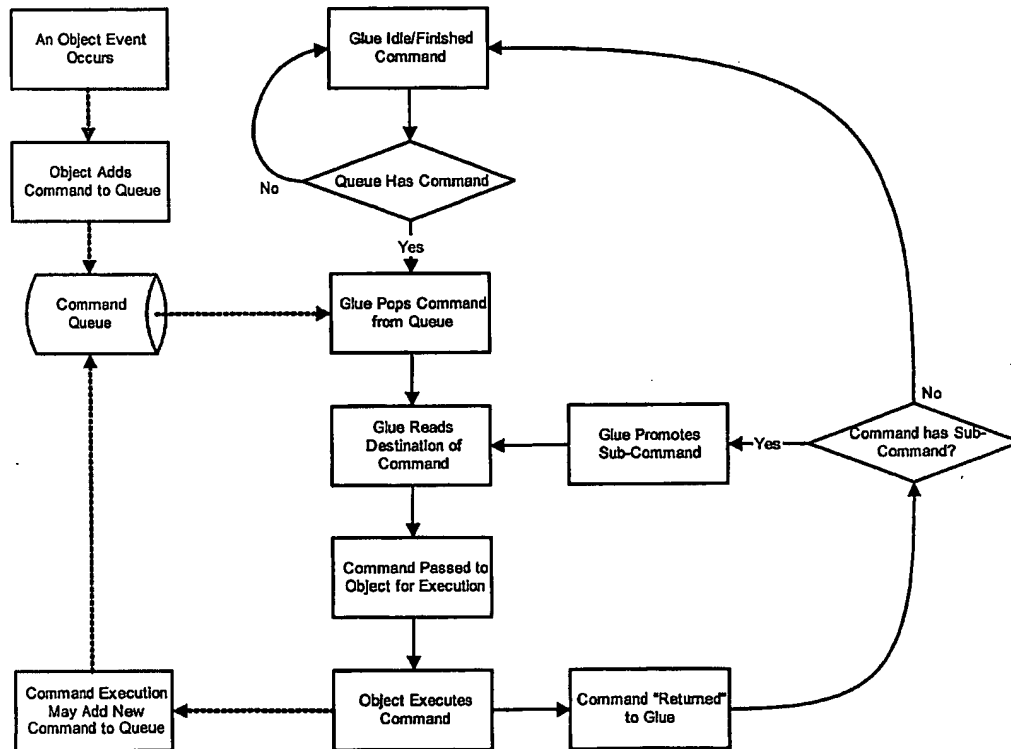
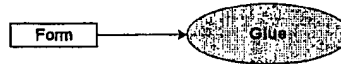
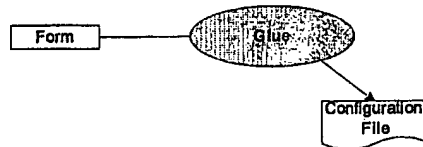


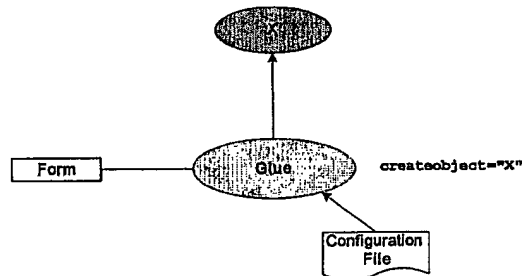
Figure 6 Process of command execution

4/23

1 Form at startup

2 Form creates Glue

3 Glue reads configuration file

4 Glue instructed to create object 'X'

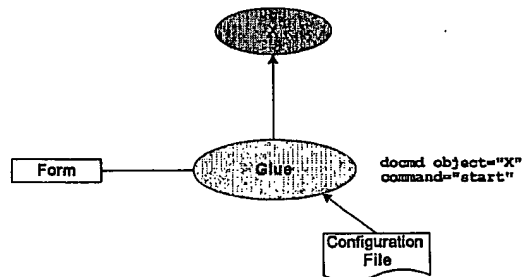
5 Glue passes start command to object 'X' for execution

Figure 7 Example of Glue executing and passing commands at startup

5/23

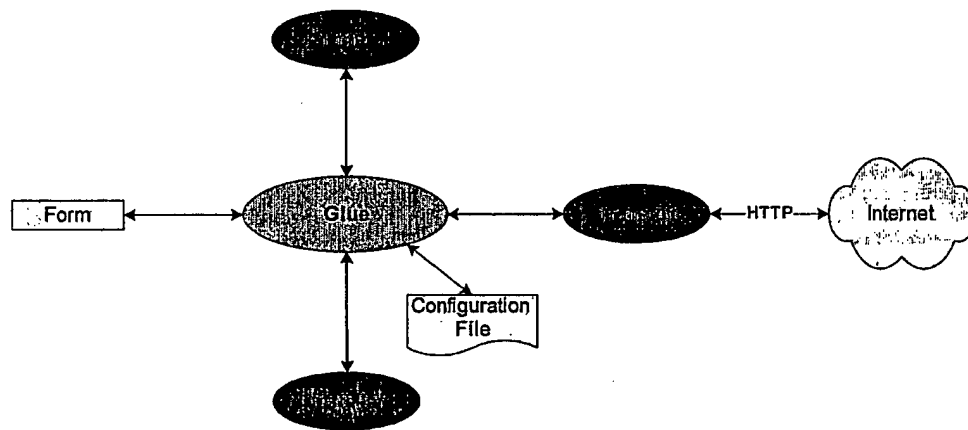


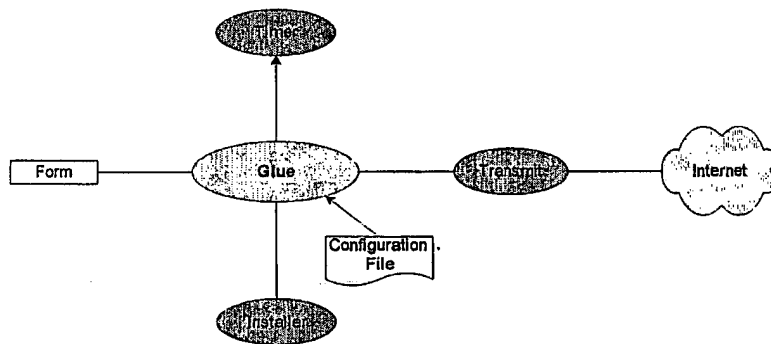
Figure 8 Base system

6/23

Figure 9 establishing a heartbeat

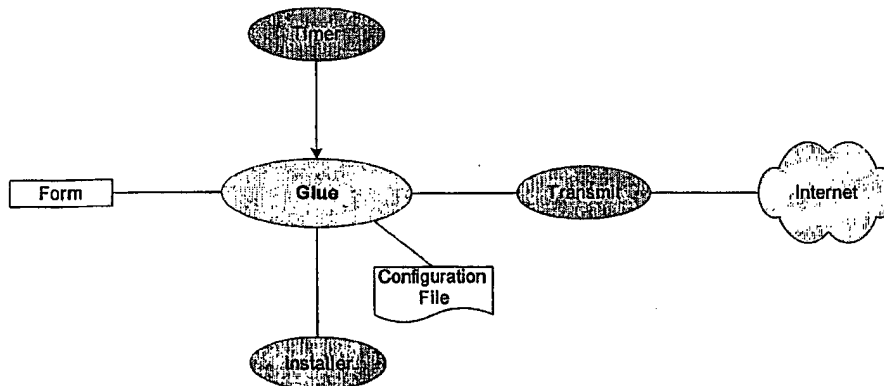
1 Timer receives instructions from Glue to send (heartbeat) command every 30 seconds

```
<docmd object="Timer" command="Register" interval="30"  
units="Seconds">  
<data type="Command">  
<docmd object="Transmit" command="HeartBeat" />  
</data>  
</docmd>
```

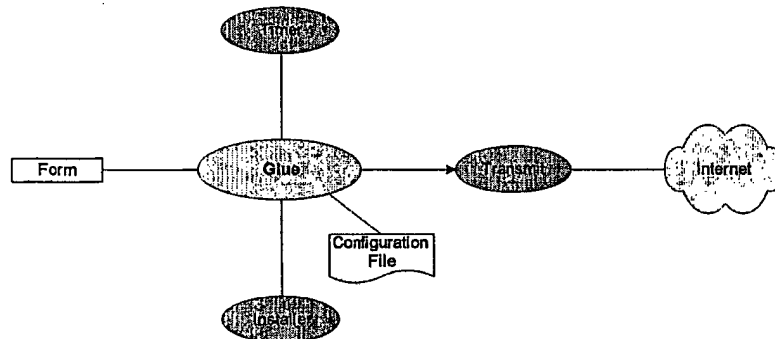
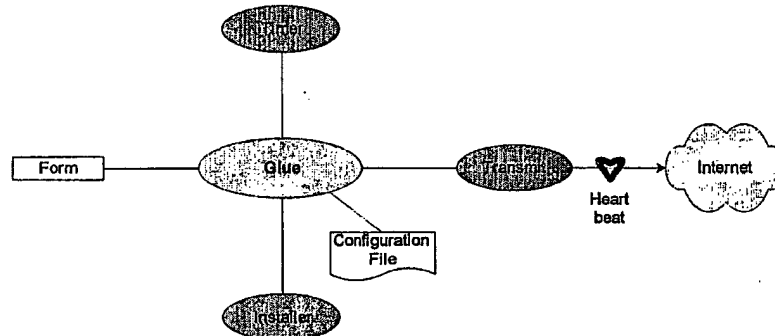
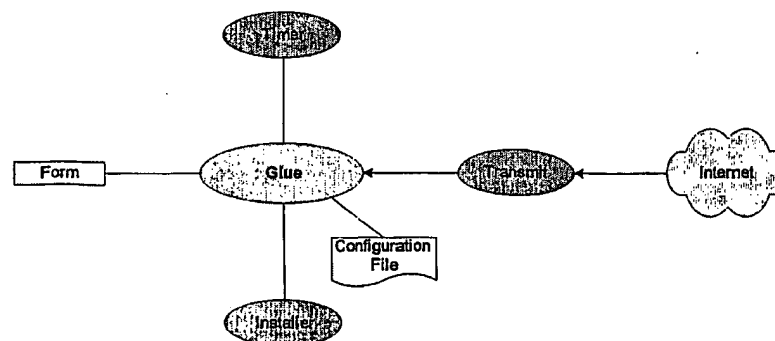


30 seconds passes

2 Timer informs Glue that there is a command to execute. Glue adds command to queue to be eventually popped off as the current command



7/23

3 Glue passes command to Transmit`<docmd object="Transmit" command="HeartBeat" />`**4 Transmit sends heartbeat****5a Heartbeat returns with "Nothing"**`<docmd object="Glue" command="Nothing" />`

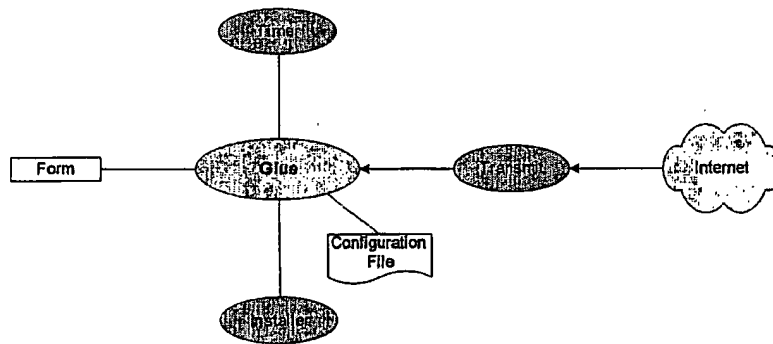
8/23

5b Heartbeat returns with commands to execute

```

<doccmd object="Installer" command="Install" filename="object.dll">
  <data type="DLL">'encoded DLL'</data>
  <data type="SuccessCommand">
    <doccmd object="Glue" command="AddSystemData">
      <doccmd object="Transmit" command="Dock" datatype="Success"/>
      <data type="Success">
        <message writeout="object.dll installed successfully."/>
      </data>
    </doccmd>
  </data>
</doccmd>
<data type="FailureCommand">
  <doccmd object="Glue" command="AddSystemData">
    <doccmd object="Transmit" command="Dock" datatype="Failure"/>
    <data type="Failure">
      <message writeout="object.dll install failed."/>
    </data>
  </doccmd>
</data>
</doccmd>

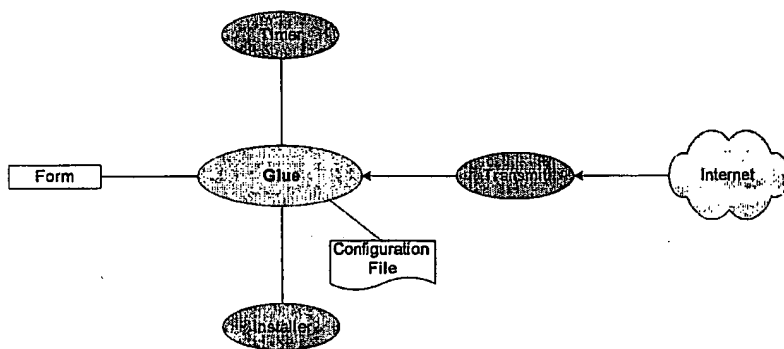
```

**Figure 9 establishing a heartbeat**

9/23

Figure 10 Installation of new object

1 Transmit receives command and passes to Glue. Glue adds command to queue to be eventually popped off as the current command



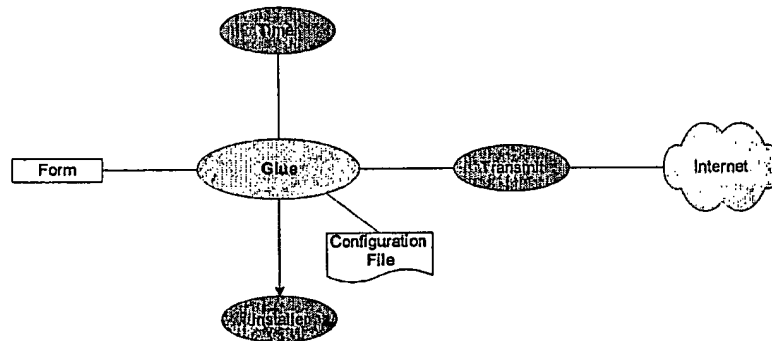
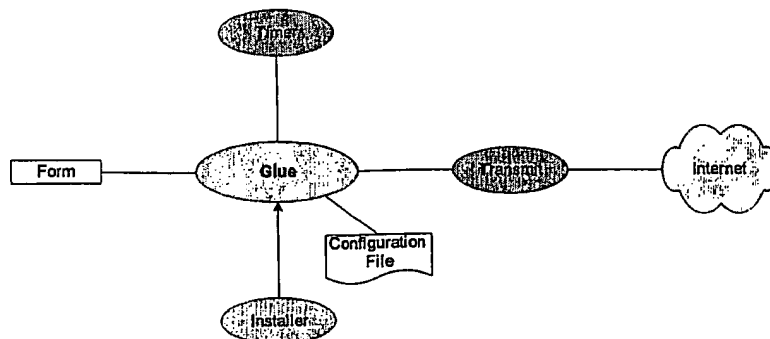
10/23

2 Glue passes command to Installer

```

<docmd object="Installer" command="Install" filename="object.dll">
  <data type="DLL">'encoded DLL'</data>
  <data type="SuccessCommand">
    <docmd object="Transmit" command="Dock" datatype="Success"/>
    <data type="Success">
      <message writeout="object.dll installed successfully."/>
    </data>
  </data>
  <data type="FailureCommand">
    <docmd object="Transmit" command="Dock" datatype="Failure"/>
    <data type="Failure">
      <message writeout="object.dll install failed."/>
    </data>
  </data>
</docmd>

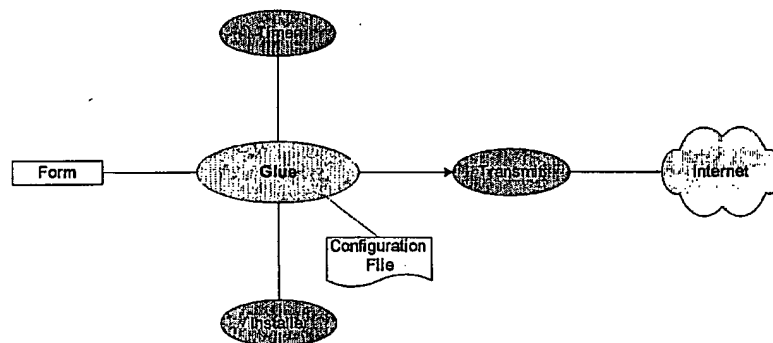
```

**3 Installer attempts to install new object. As install was successful, Installer passes the command contained within the "SuccessCommand" datatype to the queue. The current command is passed back to Glue****Figure 10 Installation of new object**

11/23

- 4 Glue completes execution of the current command. Eventually the "success" command is popped off as the current command and passed to Transmit

```
<docmd object="Transmit" command="Dock" datatype="Success"/>
<data type="Success">
  <message writeout="object.dll installed successfully."/>
</data>
```



- 5 Transmit sends status and then passes the command back to Glue. As there are no more sub-commands, Glue ceases execution and pops the next command off of the queue

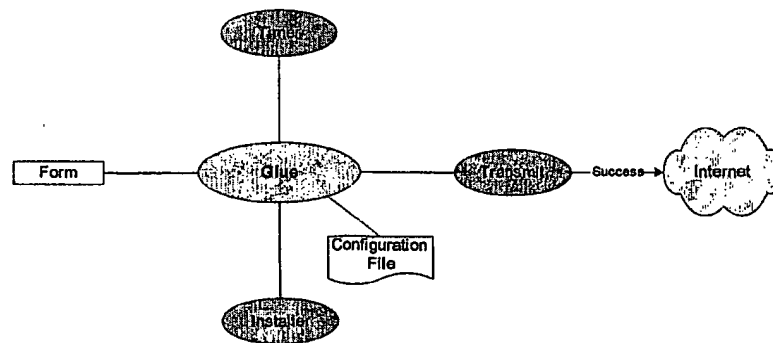


Figure 10 Installation of new object

12/23

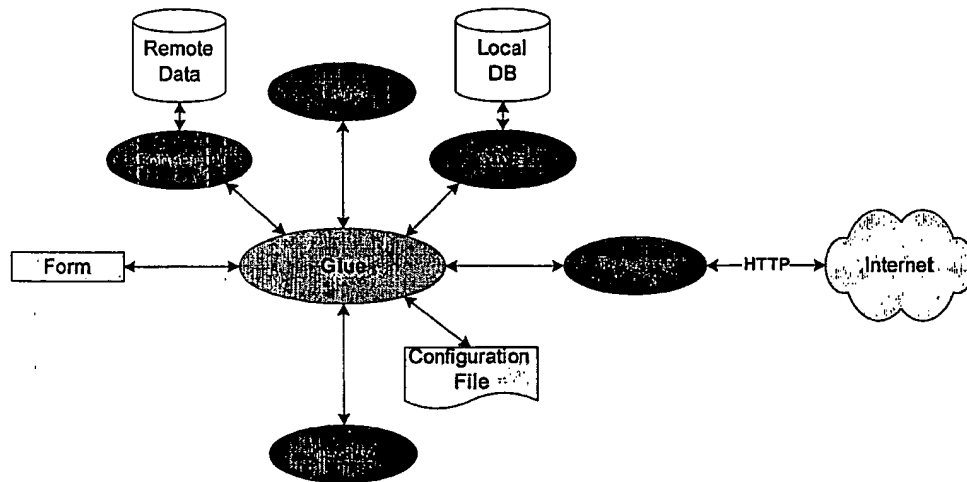


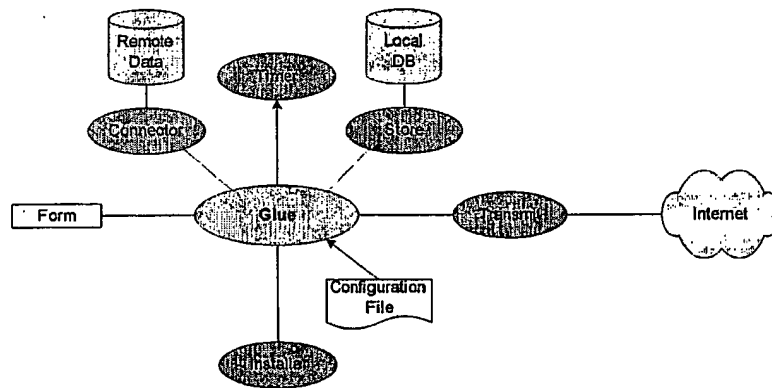
Figure 11 Functional system

13/23

Figure 12 Retrieving data at set time intervals

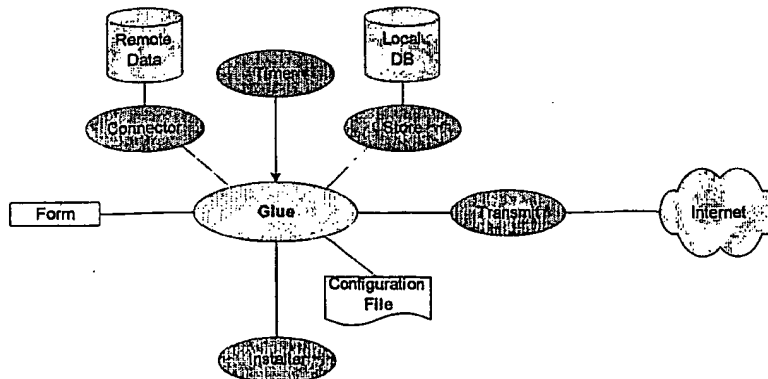
1 Timer receives instructions from Glue to send (Get-Add) command every minute

```
<doccmd object="Timer" command="Register" interval="1" units="Minutes"
synctime="00:00">
  <data type="Command">
    <doccmd object="Connector" command="GetInfo">
      <doccmd object="Store" command="AddItem" datatype="Info" />
    </doccmd>
  </data>
</doccmd>
```



1 minute passes

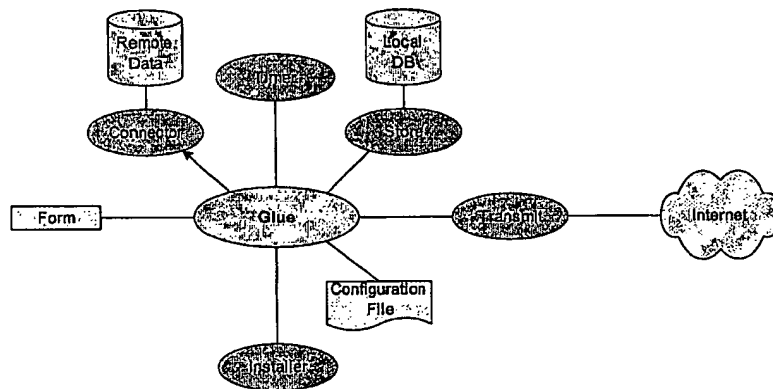
2 Timer informs Glue that there is a command to execute. Glue adds command to queue to be eventually popped off as the current command



14/23

3 Glue passes "GetInfo" command to Connector

```
<docmd object="Connector" command="GetInfo">
  <docmd object="Store" command="AddItem" datatype="Info" />
</docmd>
```



17/23

4 Connector retrieves data from remote data source and passes to Glue with current command. Glue checks if command contains sub-commands, and then promotes the first sub-command as the current command

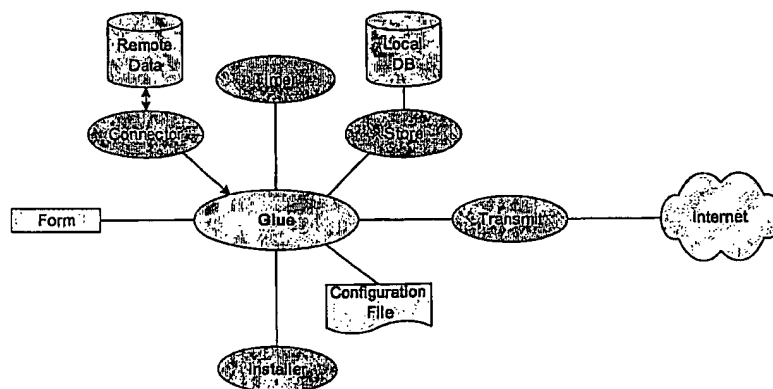
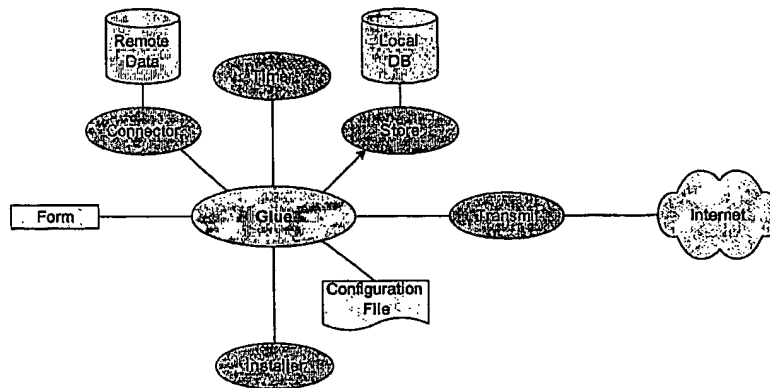


Figure 12 Retrieving data at set time intervals

15/23

5 Glue passes "AddItem" command to Store

```
<docmd object="Store" command="AddItem" datatype="Info" />
```



6 Store adds data to local database and passes message back to Glue. As there are no more sub-commands, Glue ceases execution and pops the next command off of the queue

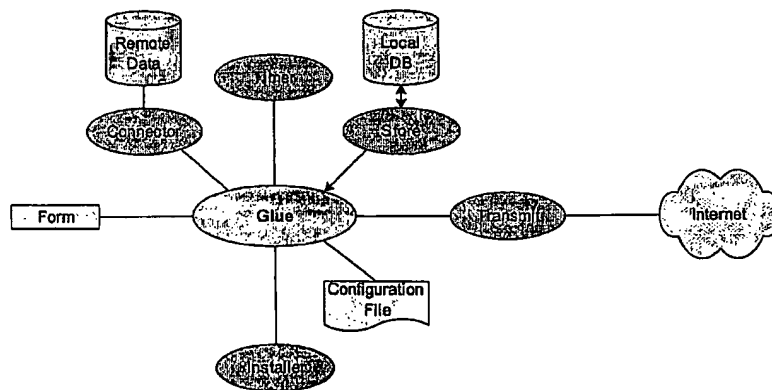


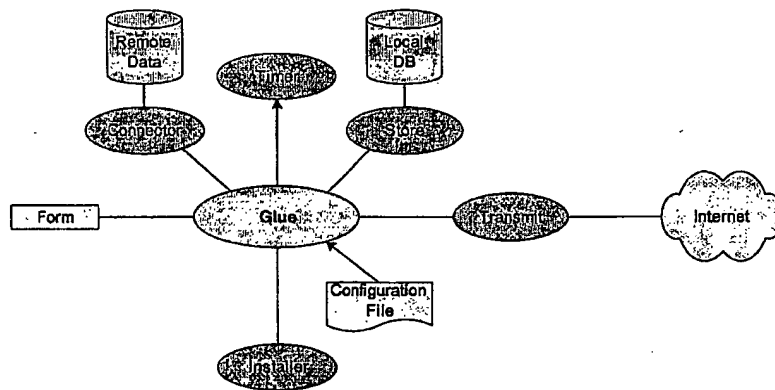
Figure 12 Retrieving data at set time intervals

16/23

Figure 13 Sending data at set time intervals

1 Timer receives instructions from Glue to send a (Bundle-Dock) command every four hours

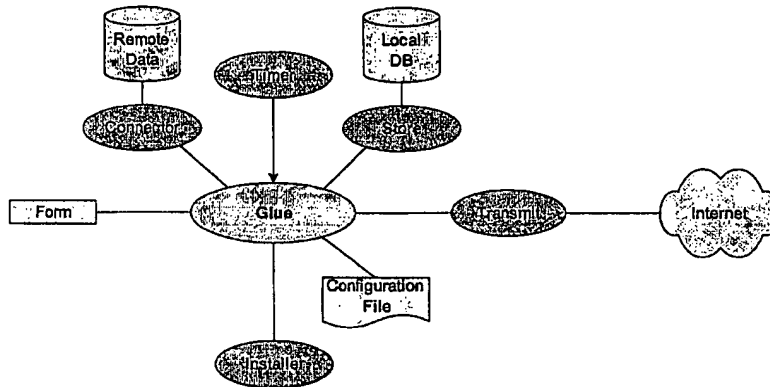
```
<docmd object="Timer" command="Register" interval="4" units="Hours"
synchtime="00:00">
  <data type="Command">
    <docmd object="Store" command="BundleCurrentItems">
      <docmd object="Transmit" command="Dock" datatype="Bundle"/>
    </docmd>
  </data>
</docmd>
```



4 hours pass

17/23

2 Timer informs Glue that there is a command to execute. Glue adds command to queue to be eventually popped off as the current command



3 Glue passes "BundleCurrentItems" command to Store

```

<docmd object="Store" command="BundleCurrentItems">
  <docmd object="Transmit" command="Dock" datatype="Bundle" />
</docmd>

```

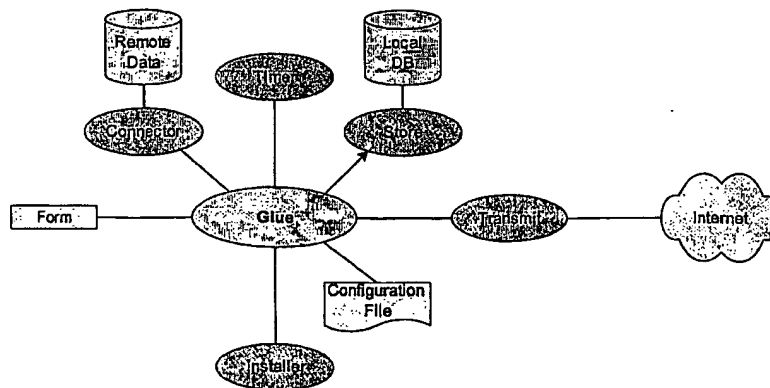
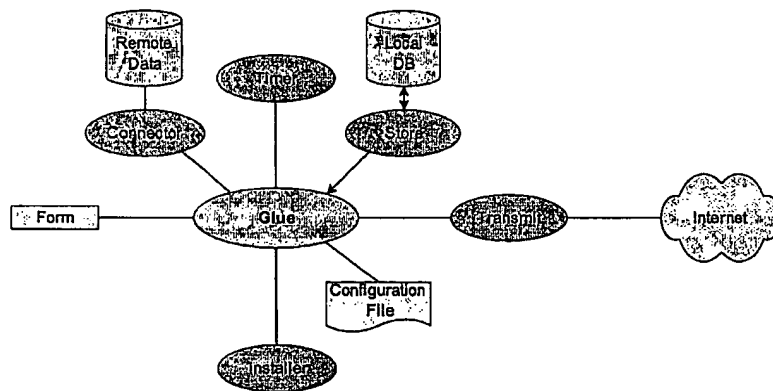


Figure 13 Sending data at set time intervals

18/23

-
- 4 Store bundles current items in the local database and passes bundle and current command back to Glue. Glue checks if command contains sub-commands, and then promotes the first sub-command as the current command



-
- 5 Glue passes "Dock" command to Transmit

```
<docmd object="Transmit" command="Dock" datatype="Bundle" />
```

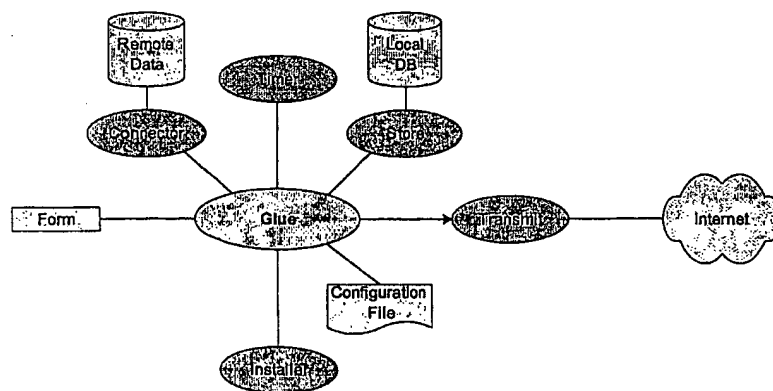
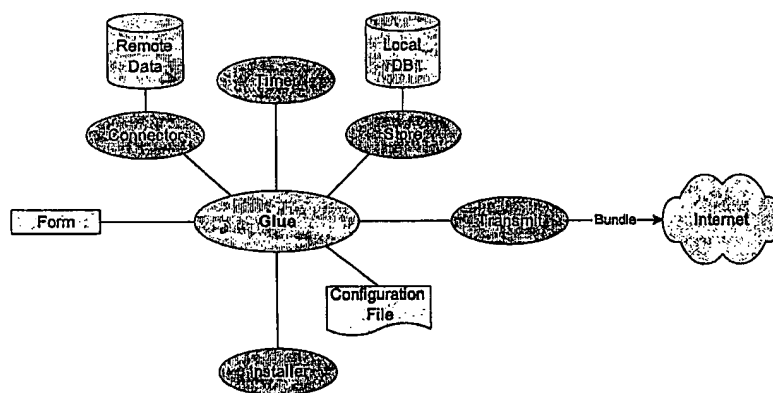


Figure 13 Sending data at set time intervals

19/23

6 Transmit sends bundle

7 Transmit returns bundle and commands to Glue. As there are no more sub-commands, Glue ceases execution and pops the next command off of the queue

Figure 13 Sending data at set time intervals

20/23

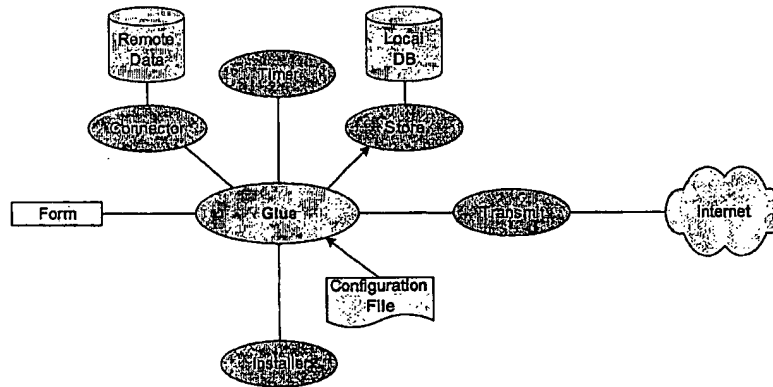
Figure 14 the sending of data bundles at set capacity levels.

1 Store receives instructions from Glue to send a (Bundle-Dock) command when local data base contains 200 data items

```

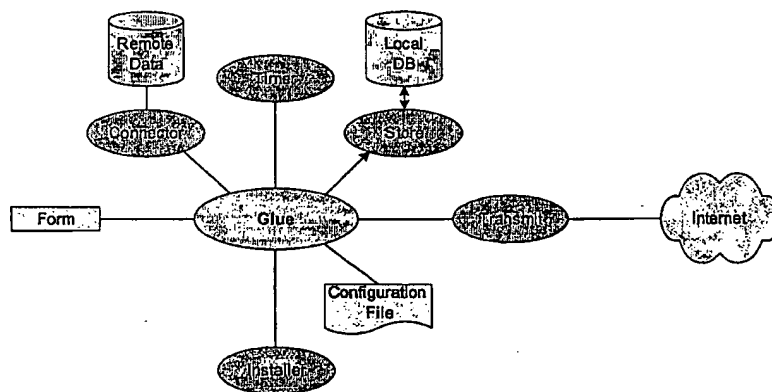
<docmd object="Store" command="SetTriggers" itemscount="200" itemsizekbs="800">
  <data type="Command">
    <docmd object="Store" command="BundleCurrentItems">
      <docmd object="Transmit" command="Dock" datatype="Bundle">
    </docmd>
  </data>
</docmd>>

```



21/23

2 200th data item is added to the local database



3 Store processes own commands first (ie. bundles) then informs Glue that there is a command to execute. Glue adds command to queue to be eventually popped off as the current command

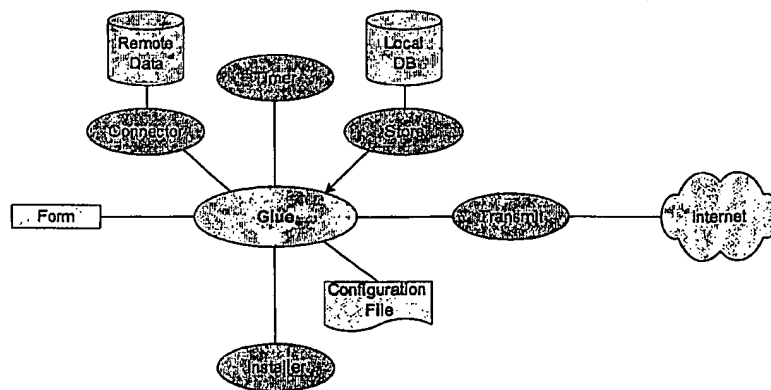
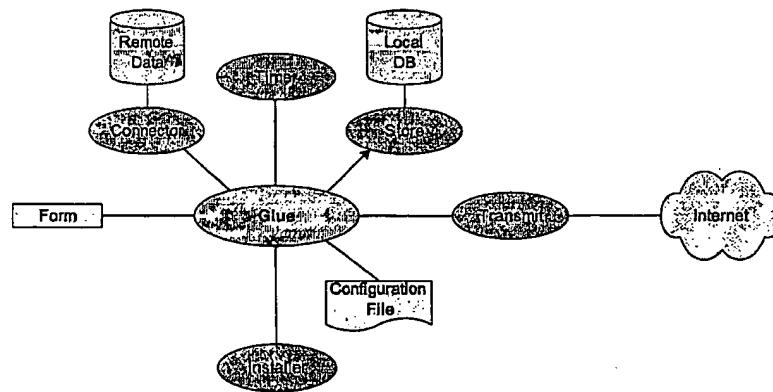


Figure 14 the sending of data bundles at set capacity levels.

22/23

4 Glue passes "BundleCurrentItems" command to Store

```
<docmd object="Store" command="BundleCurrentItems">
  <docmd object="Transmit" command="Dock" datatype="Bundle">
</docmd>
```



5 Store bundles items in local database and passes back to Glue with command. Glue checks if command contains sub-commands, and then promotes the first sub-command as the current command

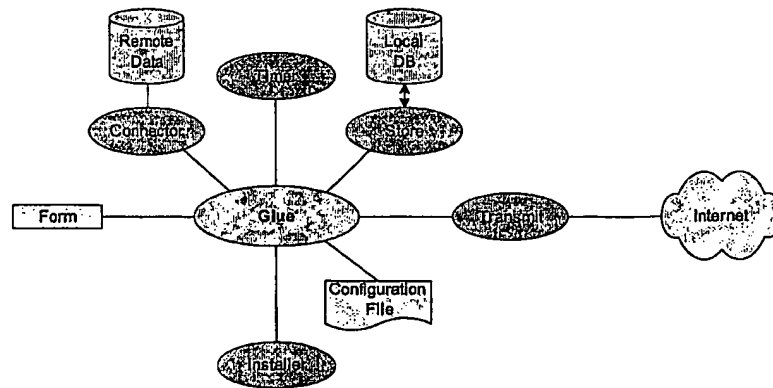


Figure 14 the sending of data bundles at set capacity levels.

23/23

6 Glue passes "Dock" command to Transmit

`<docmd object="Transmit" command="Dock" datatype="Bundle">`

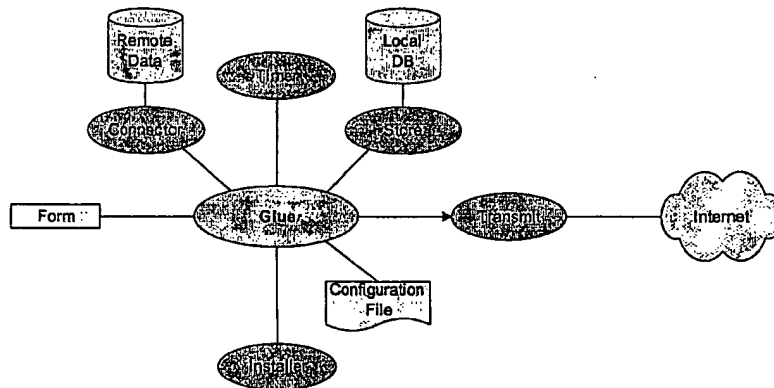
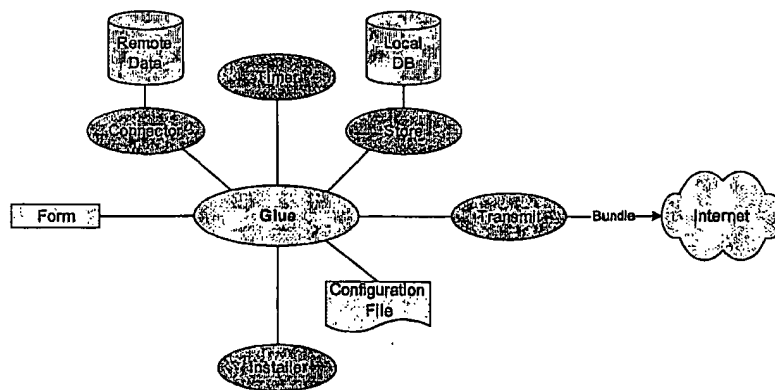
**7 Transmit sends bundle and passes command back to Glue. As there are no more sub-commands, Glue ceases execution and pops the next command off of the queue**

Figure 14 the sending of data bundles at set capacity levels.

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☒ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.